K – means clustering for Iris flower

1.0) Introduction

K-Means Clustering is an unsupervised machine learning algorithm which basically means we will just have input, not the corresponding output label. In this article, we will see it's implementation using python.

K Means Clustering tries to cluster your data into clusters based on their similarity. In this algorithm, we have to specify the number of clusters (which is a hyperparameter) we want the data to be grouped into. Hyperparameters are the variables whose value need to be set before applying value to the dataset. Hyperparameters are adjustable parameters you choose to train a model that carries out the training process itself.

2.0) K-means algorithm

The algorithm starts with an initial set of cluster centers, chosen at random or according to some heuristic procedure. In each iteration, each instance is assigned to its nearest cluster center according to the Euclidean distance between the two. Then the cluster centers are re-calculated. The center of each cluster is calculated as the mean of all the instances belonging to that cluster:

$$\mu_k = \frac{1}{N_k} \sum_{q=1|}^{N_k} x_q$$

Where Nk is the number of instances belonging to cluster k and ¹k is the mean of the cluster k. A number of convergence conditions are possible. For example, the search may stop when the partitioning error is not reduced by the relocation of the centers. This indicates that the present partition is locally optimal. Other stopping criteria can be used also such as exceeding a pre-defined number of iterations.

The algorithm is as follows:

Input: S (instance set), K (number of cluster)
Output: clusters
1: Initialize K cluster centers.
2: **while** termination condition is not satisfied **do**
3: Assign instances to the closest cluster center.
4: Update cluster centers based on the assignment.
5: **end while**

The K-means algorithm may be viewed as a gradient-decent procedure, which begins with an initial set of K cluster-centers and iteratively updates it so as to decrease the error function. A rigorous proof of the finite convergence of the K-means type algorithms is given in (Selim and Ismail, 1984). The complexity of T iterations of the K-means algorithm performed on a sample size of m instances, each characterized by N attributes, is: O(T * K * m * N). This linear complexity is one of the reasons for the popularity of the K-means algorithms.

3.0) Application of k-means for classification of the Iris flower

3.1) Project Description

This project is based on the classification of species of the Iris flower. The Iris flower has 3 species: The Iris setosa, Iris virginica and Iris versicolor. This project will use the k-means clustering algorithm to classify a data set containing these 3 species of flowers and give a classification report on the precision, recall, f-score and support of the output.

3.2) Dataset

The dataset of this project is gotten from the UCI machine learning database. The dataset can be accessed at https://archive.ics.uci.edu/ml/machine-learning-databases/iris

The dataset contains over 150 data items of 3 species of the iris flower with 4 properties. These properties are: length, width, sepals and petals.

3.3) Structure of the code

First we import the necessary libraries for the proper running of the code.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import classification_report
```

- Clustering of unlabeled data can be performed with the help of sklearn.cluster module. From this module, we can import the KMeans package.
- Pandas is used for reading and writing spreadsheets
- Numpy is used for carrying out efficient computations
- Matplotlib is used for visualization of data

Next we import the iris.csv dataset for the data and access columns 1 to 4 which contain the 4 attributes of each flower in the dataset.

```
# Importing the datasets
dataset = pd.read_csv('iris2.csv')
X = dataset.iloc[:,[0,1,2,3]].values
dataset['Species'] = pd.Categorical(dataset["Species"])
dataset["Species"] = dataset["Species"].cat.codes
```

The *datset["species"].cat.codes* function is used to normalize the 5^{th} column of the dataset which contains names of species as a string. This is important because the k-means algorithm only works with numerical data, so normalization of the data to numeric is necessary.

Next we determine the optimum number of clusters (the value of k) which can be used in this clustering process. This is done graphically through the application of the "elbow method". With the elbow method, the points at the elbow of the curve shows the optimum point from which the value of k can be determined. This is done as follows:

```
#optimum number of clusters for K-Means classification
from sklearn.cluster  import KMeans
wcss = []
for i in range(1,11):
    kmeans= KMeans(n_clusters= i, init='k-means++', max_iter=400, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,11),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

The elbow method is implemented using the *"sklearn"* library. It plots a graph of *wcss* on the y-axis against number of clusters on the x-axis. By using the *plt.plot* method, we can display the graph to the screen.

Next we apply the k-means algorithm using an optimum k value of 3 with a maximum of 400 iterations.

```
# Applyig Kmeans to the dataset
kmeans = KMeans(n_clusters= 3, init='k-means++', max_iter=400, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)
target_names = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
cat = classification_report(dataset['Species'],kmeans.labels_,target_names=target_names)
# Visualizing
plt.scatter(X[y_kmeans==0,0],X[y_kmeans==0,1],s = 100, c = 'red', label = 'Iris-setosa')
plt.scatter(X[y_kmeans==1,0],X[y_kmeans==1,1],s = 100, c = 'blue', label = 'Iris-versicolour')
plt.scatter(X[y_kmeans==2,0],X[y_kmeans==2,1],s = 100, c = 'green', label = 'Iris-virginica')
```

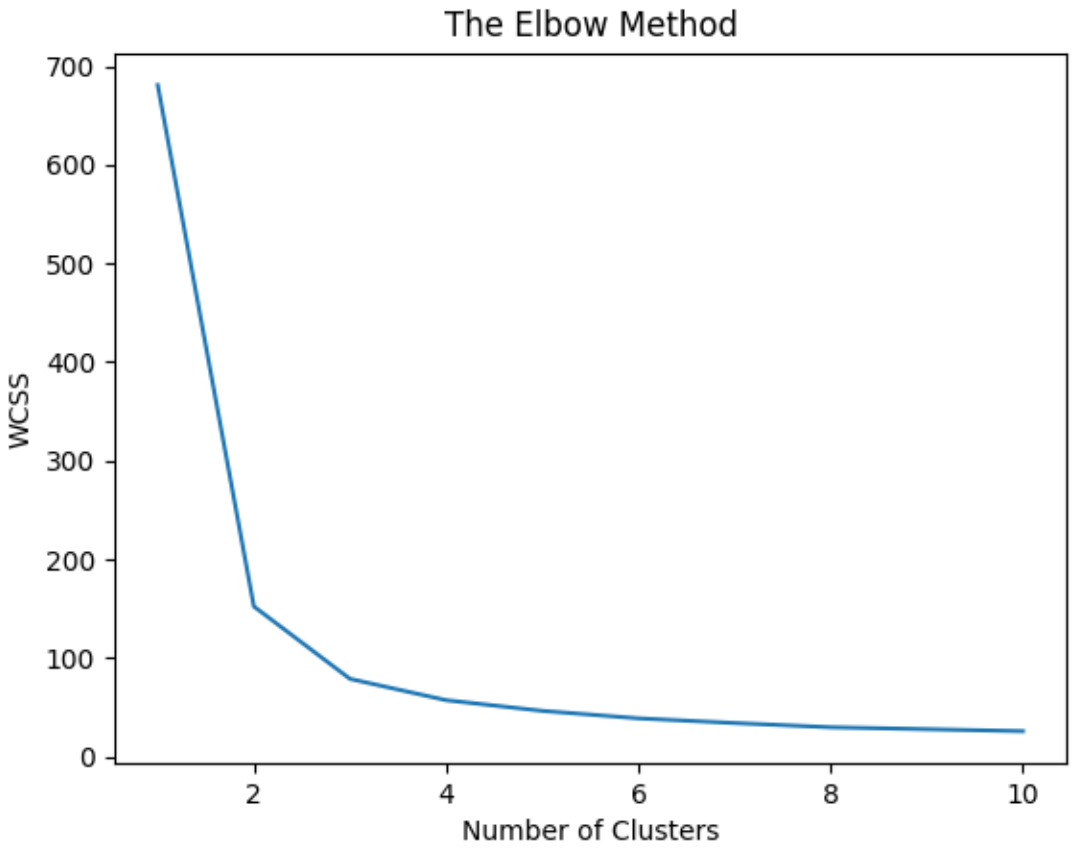
The function *y_kmeans* is used to predict the best cluster by calculating the Euclidean distance. This is applied to all the 3 centroids to verify which cluster is the shortest (best) cluster for that data point. This is then displayed to the screen using different colors. The Iris-setosa are represented by red, the Iris-versicolour are represented by blue and the Iris-virginica are represented by green.

Finally we display the position of the 3 centroids both on the graph (represented by the yellow color) and on the console, and the classification report which consist of the precision, recall, f-score and support.

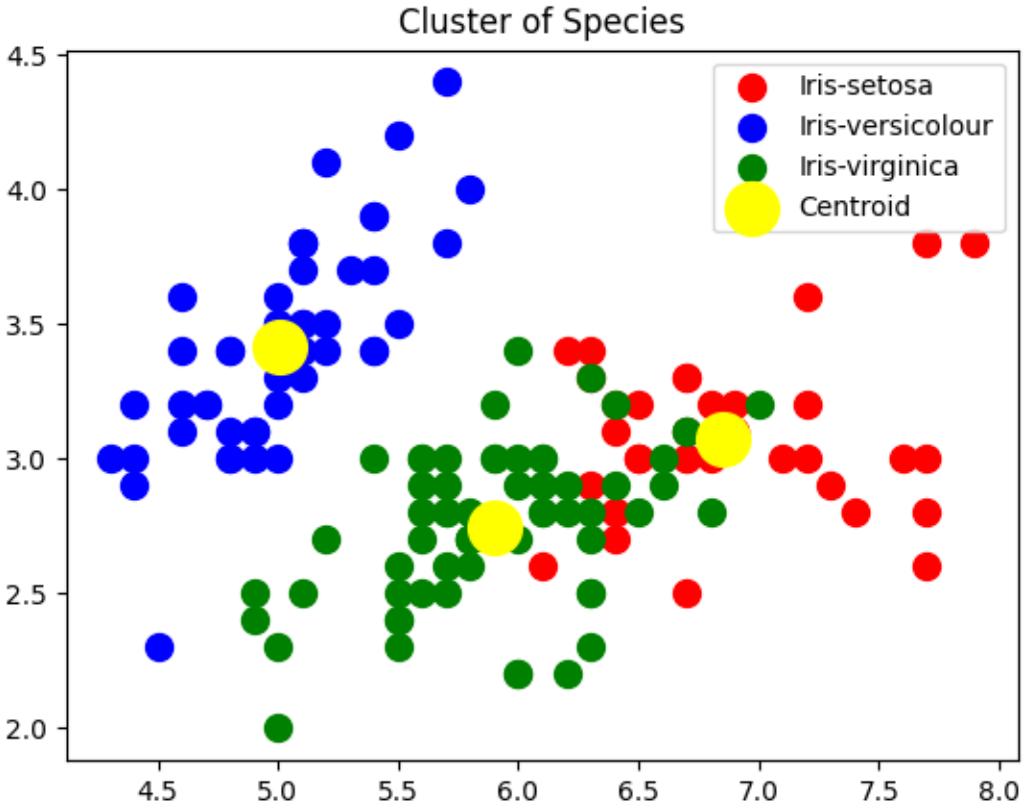
```
#Cluster Centroid
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1] ,s = 400, c = 'yellow', label = 'Centroid')
plt.title('Cluster of Species')
plt.legend()
plt.show()

#Display kmeans cluster positions and the classification report
print (kmeans.cluster_centers_)
print (cat)
```

3.4) Results

The graph which for calculating the optimum k value for classification using the elbow method is shown below.



The clusters are shown in the diagram below



The position of the clusters and the classification data displayed to the console is as shown below.

```
C:\Users\lionel\PycharmProjects\PracticeExercises\venv\Scripts\python.ex
[[6.85       3.07368421 5.74210526 2.07105263]
 [5.006      3.418      1.464      0.244     ]
 [5.9016129  2.7483871  4.39354839 1.43387097]]
                precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00        50
Iris-versicolor       0.00      0.00      0.00        50
 Iris-virginica       0.23      0.28      0.25        50

       accuracy                           0.09       150
      macro avg       0.08      0.09      0.08       150
   weighted avg       0.08      0.09      0.08       150


Process finished with exit code 0
```

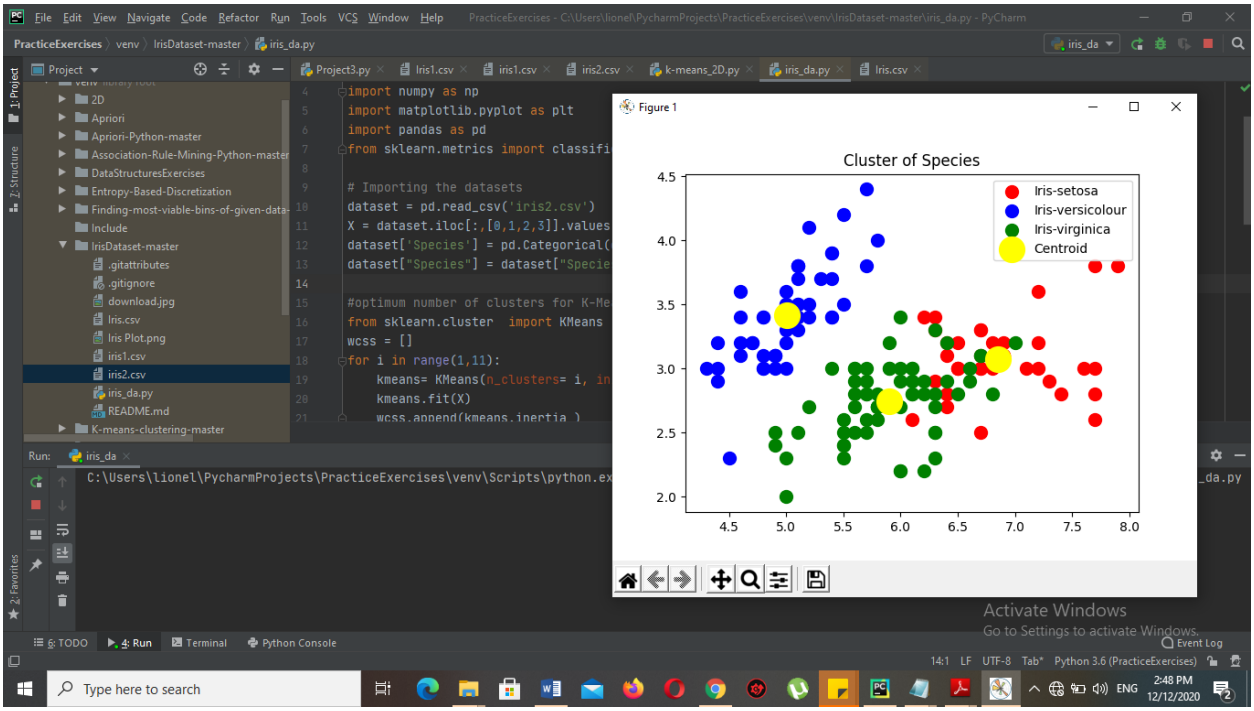


Figure 1
Cluster of Species
Iris-setosa
Iris-versicolour
Iris-virginica
Centroid
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Activate Windows
Go to Settings to activate Windows.